WHAT IS CLAIMED IS:

1. A particularly configurable microprocessor for processing computer programs which are selectively operable on said particularly configurable microprocessor, comprising:

an instruction decoder including a plurality of buffers for receiving instructions, said instruction decoder being programmable so that it accepts instruction op codes in excess of a minimal set of instruction op codes required for program execution and storing plural answers in selected ones of the plurality of buffers; and

logic circuitry for processing op codes received by said instruction decoder, said logic circuitry configured to process the accepted op codes including excess op codes to produce substantive and obfuscating results.

2. The microprocessor of claim 1, wherein:

a plurality of reconfigurable logic gates calculate results of execution of an instruction; and

the logic gates calculate the results of the execution of an instruction and make provisions for outputting correct results along with plausible wrong results.

3. The microprocessor of claim 2, wherein said logic gates include provisions for accepting correct data operands and plausible wrong data operands.

4. The microprocessor of claim 2, further comprising a capability of accepting a key shared by with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes.

5. The microprocessor of claim 1, further comprising:

a capability of accepting a key shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes; and

an output register for data results able to contain both correct results and plausible wrong results, the results in word locations in the output register coordinated by the key.

6. The microprocessor of claim 1, wherein:

program instructions are provided in a pipeline architecture;

an information key establishes instruction security commands at a plurality of steps in said pipeline architecture; and

an arithmetic logic unit (ALU) provides variability of logic circuitry for execution of encrypted op codes or standard op codes that provide standard instruction operation types.

7. The microprocessor of claim 6, wherein:

a key is shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes;

the key is stored in more than one memory cell type including a Read Only Memory (ROM), an Electrically Erasable Programmable Read Only Memory (E^2PROM), and a Random Access Memory (RAM), the key including bits optionally expanded into a larger set of bits which control the instruction decoder, signal routing, and logic gate reconfiguration;

a serial number in ROM participates in the allocation of logic gates and routing of signals, and communicated to the compiler to inform the compiler of custom allocation and routing; and

the key provides a capability of controlling signal routing, and logic gate reconfiguration whether the op codes are encrypted op codes or standard op codes.

8. The microprocessor of claim 7, further comprising an output register for data results able to contain both correct results and plausible wrong results which are in word locations in the output register coordinated by the key.

9. The microprocessor of claim 8, wherein:

a plurality of reconfigurable logic gates calculate results of execution of an instruction;

said plurality of the logic gates include provisions for accepting correct data operands and plausible wrong data operands; and

said plurality of the logic gates include provisions for outputting correct results along with plausible wrong results.

10. The microprocessor of claim 9, wherein the output register for data results contains both correct results and plausible wrong results which are in word locations in the output register, the locations of the results coordinated by the key.

11. The microprocessor of claim 10, wherein"

a plurality of the memory locations are dispersed within a layout;

a plurality of reconfigurable logic gates are able to calculate results of execution of an instruction;

said plurality of the logic gates include provisions for accepting correct data operands and plausible wrong data operands; and

said plurality of the logic gates include provisions for outputting correct results along with plausible wrong results.

12. The microprocessor of claim 11, further comprising:

the key providing a capability of re-allocating memory resources and register resources;

a serial number in ROM which participates in the allocation of logic gates and routing of signals; and

the serial number used in combination with the key in providing said capability.

13. The microprocessor of claim 1, further comprising:

a capability of accepting a key shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes; and

variations of data numeric representations coordinated by the key and the encrypted op codes.

14. The microprocessor of claim 1, further comprising:

a capability of accepting a key shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes; and

an instruction buffer which contains logic which can route a subset of the instruction bits from bit location in the buffer to destination logic gates which reach the

programmable instruction decoder and said instruction buffer interdependency checking logic block.

15. The microprocessor of claim 1, further comprising:

logic gates configured to process data coded in various numeric representations and the logic gates able to accept results of the instruction execution using various numeric representations;

logic gates configured to immediately process said coded data; and

the data representation able to change several times during the execution of a program so that numeric encodings of input data operands and output data results can vary.

16. The microprocessor of claim 15, further comprising:

a capability of accepting a key shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes; and

the variations of the data numeric representations coordinated by means of the key and the encrypted op codes.

17. The microprocessor of claim 1, wherein:

program instructions are provided in a pipeline architecture;

information keys are established as instruction security commands at a plurality of steps in said pipeline architecture; and

an arithmetic logic unit (ALU) provides variability of logic circuitry for execution of encrypted op codes or standard op codes that provide standard instruction operation types.

18. The microprocessor of claim 1, wherein the instruction decoder provides plural answers for storage in the plurality of buffers, and the excess op codes provide plausible wrong answers.

19. The microprocessor of claim 1, further comprising:

a capability of accepting a key shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes; and

data and instructions provided to a computer via program information includes an intentional introduction of errors which are correctable with error correction algorithms, said correction algorithms pre-selected according to the key.

20. The microprocessor of claim 19, further comprising:

an instruction buffer which contains logic which can route a subset of the instruction bits from bit location in the buffer to destination logic gates which eventually reach a programmable instruction decoder and an instruction buffer interdependency checking logic block; and

said correction algorithms pre-selected according to long instruction words and changed on a periodic basis by codes provided in the instructions gathered into the instruction buffer.

21. The microprocessor of claim 19, wherein the instruction buffer interdependency checking logic includes any combination of the following:

multiplexers to select a subset of bits from a long instruction word in the instruction buffer to be logically combined to match a sequencer value;

a sequencer incremented at times determined by the key and which is reset upon the occurrence of the sequencer reset code in the instruction buffer;

distribution of bits for one encrypted op code across several long instruction words in the instruction buffer;

distribution of several encrypted op codes around the long instruction words in the instruction buffer;

a program counter which does not normally increment by one, but which increments by some other constant or variable amount determined by the serial number, the key, and the sequencer value so that encrypted op codes which will be used sequentially in time do not occur sequentially in the instruction buffer, and for which, the time sequential chosen op codes are selected by the multiplexer controlled by the key, the serial number, and the sequencer;

error correction circuits controlled by the key, sequencer, and supplementary error correcting codes received from the instruction buffer by means of the multiplexers; and

dependency validation codes received through the multiplexer of the instruction buffer checked by logic circuits that depend on the key, the serial number, instruction bits, and camouflage bits.

22. The microprocessor of claim 21, wherein dependency validation codes are received through the multiplexer of the instruction buffer checked by logic circuits that depend on the key, the serial number, instruction bits, and camouflage bits so that incorrect validation bits provide an alarm.

23. The microprocessor of claim 1, further comprising:

a store for a key shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes; and

dependency validation codes received through an instruction buffer checked by logic circuits that depend on the key, a serial number, instruction bits, and camouflage bits so that incorrect validation bits provide an alarm.

24. The microprocessor of claim 1, further comprising:

a plurality of storage locations for keys, with the keys further determining storage locations of satellite keys and satellite access flags, said locations intentionally varied; and

key-dependent storage of remote access approval flags, the remote access approval flags encoded so as to obscure the locations of said approval flags.

25. The microprocessor of claim 24, further comprising logic circuitry for requiring network handshaking, the network handshaking further used to provide additional key information for continued operation.

26. Method for processing computer programs selectively operable on one or more selected individual processors, comprising:

programming an instruction decoder to accept instruction op codes in excess of a set of instruction op codes required for execution of a program;

providing plural answers from the instruction decoder, including plausible wrong answers; and

selecting a predetermined buffer, thereby permitting further operation with a selected one of the plurality of answers.

27. The method of claim 26, further comprising:

using reconfigurable logic gates for calculating the results of execution of an instruction, the calculation of results of the execution of an instruction including accepting correct data operands and plausible wrong data operands; and

outputting correct results along with plausible wrong results.

28. The method of claim 27, further comprising:

using at least a portion of the reconfigurable logic gates for calculating the results of the execution of an instruction;

using said portion of the logic gates for accepting correct data operands and plausible wrong data operands; and

using said portion of the logic gates for outputting correct results along with plausible wrong results.

29. The method of claim 28, further comprising:

providing a key shared with a compiler;

encrypting standard op codes with the compiler using the key;

storing the key in more than one memory cell type, including a Read Only Memory (ROM), an Electrically Erasable Programmable Read Only Memory (E^2PROM), and a Random Access Memory (RAM);

expanding key bits in the key into a larger set of bits which control the instruction decoder, signal routing, and logic gate reconfiguration;

using a serial number in ROM in the allocation of logic gates and routing of signals, the serial number communicated to the compiler to inform the compiler of custom allocation and routing; and

using the key for signal routing, and logic gate reconfiguration whether the op codes are encrypted op codes or standard op codes.

30. The method of claim 26, further comprising:

using reconfigurable logic gates for calculating the results of the execution of an instruction;

accepting correct data operands and plausible wrong data operands; and

outputting correct results along with plausible wrong results.

31. The method of claim 26, further comprising:

providing a key shared with a compiler;

encrypting standard op codes with the compiler using the key;

providing correct results and plausible wrong results in an output register; and

coordinating the results in word locations in an output register according to the key.

32. The method of claim 26, further comprising:

providing program instructions in a pipeline architecture; and

establishing information keys as instruction security commands at a plurality of steps in said pipeline architecture, wherein an arithmetic logic unit (ALU) provides variability of logic circuitry for execution of encrypted op codes or standard op codes that provide standard instruction operation types.

33. The method of claim 26, further comprising:

providing a key shared with a compiler, the key used by the compiler to encrypt standard op codes into encrypted op codes; and

using the key to coordinate the variations of the data numeric representations and the encrypted op codes.

34. The method of claim 33, further comprising using the key to provide a capability of re-allocating memory resources and register resources.

35. The method of claim 40, further comprising:

using a serial number in combination with the key in providing said capability; and

using the reallocation of memory and register resources whether the op codes are encrypted or not encrypted.

36. The method of claim 26, further comprising routing a subset of op codes through an instruction buffer to destination logic gates, which reach a programmable instruction decoder and an instruction interdependency checking logic block.

37. The method of claim 26, further comprising changing the programming of the instruction decoder during the execution of a program so that the numeric encodings of input data operands and output data results will change.

38. The method of claim 26, further comprising:

using reconfigurable logic gates able to calculate results of execution of an instruction; and

reconfiguring the logic gates outputting correct results from the logic gates along with plausible wrong results.

39. The method of claim 26, further comprising:

providing a key shared with a compiler;

encrypting standard op codes with the compiler using the key;

providing data containing both correct results and plausible wrong results at an output register; and

providing the correct results in word locations in the output register coordinated by the key.

40. The method of claim 26, further comprising:

using logic for requiring network handshaking; and

further using the network handshaking to provide additional key information for continued operation.

41. Method of compiling a computer program for use on a selected processor, comprising:

providing instruction op codes in excess of a set of instruction op codes required for execution of a program;

providing instruction op codes to provide plural answers from the instruction decoder, including plausible wrong answers; and

providing instruction op codes to select a predetermined buffer, thereby selecting one of the plurality of answers.

42. The method of claim 41, further comprising:

providing a key shared with a compiler; and

encrypting standard op codes with the compiler using the key;

establishing word locations in an output register according to the key.

43. The method of claim 42, further comprising using a serial number in combination with the key.